CLAIMS

1. An apparatus comprising:

at least one processor;

a memory coupled to the at least one processor;

an object oriented program residing in the memory comprising a plurality of instructions; and

a dynamic compiler residing in the memory and executed by the at least one processor, the dynamic compiler being invoked during execution of the object oriented program, the dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object.

2. The apparatus of claim 1 wherein the dynamic compiler comprises:

an escape analysis mechanism that marks each instruction that allocates a new object as one of global escape, no escape, and arg escape based on information available from previously-loaded classes that are part of the object oriented program; and

an object allocation mechanism that allocates at least one object that is created by an instruction marked as no escape by the escape analysis mechanism to an invocation stack frame for a method that allocates the object.

3. The apparatus of claim 2 wherein the dynamic compiler analyzes each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object by the object allocation mechanism to the invocation stack frame, and if so, the dynamic compiler changes the allocation of the object to a heap.

4. The apparatus of claim 1 wherein the dynamic compiler changes the allocation of the object from the invocation stack frame to a heap due to information that becomes available from at least one class that is loaded after the dynamic compiler allocates the at least one object to the invocation stack frame.

5. The apparatus of claim 4 wherein the dynamic compiler changes at least one pointer to the object allocated on the invocation stack to point to an object allocated on the heap as a result of information that becomes available as more classes that are part of the object oriented program are loaded.

6. An apparatus comprising:

at least one processor;

a memory coupled to the at least one processor;

an object oriented program residing in the memory comprising a plurality of instructions;

a portion of the object oriented program that is selected for dynamic compilation;

a dynamic compiler residing in the memory and executed by the at least one processor, the dynamic compiler being invoked during execution of the object oriented program, the dynamic compiler comprising:

an escape analysis mechanism that marks each instruction that allocates a new object as one of global escape, no escape, and arg escape based on information available from classes that are part of the object oriented program that have been previously loaded at run-time; and

an object allocation mechanism that allocates at least one object that is created by an instruction marked as no escape by the escape analysis mechanism to an invocation stack frame for a method that allocates the object.

7. A method for allocating objects to memory in an object oriented program during dynamic compilation of a portion of the object oriented program while the object oriented program is executing, the method comprising:

(A) determining whether compilation of the portion is needed;

(B) if compilation of the portion is needed:

(B1) analyzing each instruction in the portion that allocates a new object; and

(B2) allocating at least one object that is created by an instruction to an invocation stack frame for a method that allocates the at least one object.

8. The method of claim 7 wherein act (A) comprises determining whether a method in the portion has been executed a number of times equal to or greater than a predetermined threshold value.

9. The method of claim 7 wherein acts (B1) and (B2) comprise:

marking each instruction in the portion that allocates a new object as one of global escape, no escape, and arg escape based on information available from classes that are part of the object oriented program that have been previously loaded at run-time; and

allocating at least one object that is created by an instruction marked as no escape by the escape analysis mechanism to an invocation stack frame for a method that allocates the at least one object.

10. A method for allocating objects in an object oriented program to memory, the method comprising:

loading a plurality of classes that are part of the object oriented program;

executing code from at least one of the plurality of loaded classes;

determining whether dynamic compilation of a portion of the object oriented program is needed;

if dynamic compilation of the portion is needed, allocating at least one object to an invocation stack frame for a method that allocates the at least one object.

11. The method of claim 10 further comprising:

analyzing compiled code as each subsequent class in the object oriented program is loaded; and

changing the allocation of the at least one object from the invocation stack frame to a heap.

12. A method for executing an object oriented program, the method comprising:

loading a plurality of classes;

executing code from the plurality of classes in an interpreted mode;

monitoring execution frequency of methods in the plurality of classes;

when execution frequency for a method exceeds a predetermined threshold value, dynamically compiling a portion of the plurality of classes that includes the method, the dynamic compiling allocating an object to an invocation stack frame for a method that allocates the object if the lifetime of the object in the plurality of loaded classes does not escape the method; and

executing the compiled portion.

13. A program product comprising:

a dynamic compiler that is invoked during execution of an object oriented program, the dynamic compiler allocating at least one object in the object oriented program to an invocation stack frame for a method that allocates the at least one object; and

signal bearing media bearing the dynamic compiler.

14. The program product of claim 13 wherein the signal bearing media comprises recordable media.

15. The program product of claim 13 wherein the signal bearing media comprises transmission media.

16. The program product of claim 13 wherein the dynamic compiler comprises:

an escape analysis mechanism that marks each instruction that allocates a new object as one of global escape, no escape, and arg escape based on information available from previously-loaded classes that are part of the object oriented program; and

an object allocation mechanism that allocates at least one object that is created by an instruction marked as no escape by the escape analysis mechanism to an invocation stack frame for a method that allocates the at least one object.

17. The program product of claim 16 wherein the dynamic compiler analyzes each class as it is loaded to determine whether the newly-loaded class affects the allocation of an object by the object allocation mechanism to the invocation stack frame, and if so, the dynamic compiler changes the allocation of the object to a heap.

18. A program product comprising:

(A) a dynamic compiler that compiles a portion of an object oriented program, the dynamic compiler being invoked during the execution of the object oriented program, the dynamic compiler comprising:

(A1) an escape analysis mechanism that marks each instruction that allocates a new object as one of global escape, no escape, and arg escape based on information available from classes that are part of the object oriented program that have been previously loaded at run-time;

(A2) an object allocation mechanism that allocates at least one object that is created by an instruction marked as no escape by the escape analysis mechanism to an invocation stack frame for a method that allocates the object; and

(B) signal bearing media bearing the dynamic compiler.

19. The program product of claim 18 wherein said signal bearing media comprises recordable media.

20. The program product of claim 18 wherein said signal bearing media comprises transmission media.

*****